Live User-Guided Intrinsic Video for Static Scenes

Abhimitra Meka*, Gereon Fox*, Michael Zollhöfer, Christian Richardt and Christian Theobalt, *Member, IEEE*

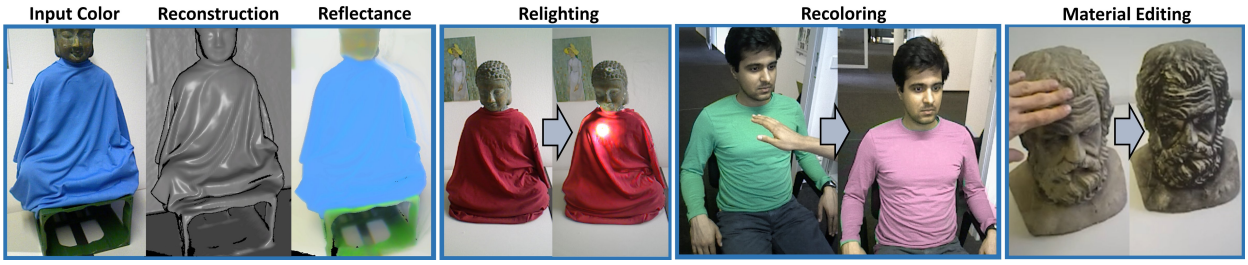

Fig. 1. We propose a novel approach for live, user-guided intrinsic video decomposition. We first obtain a dense volumetric reconstruction of the scene using a commodity RGB-D sensor. The reconstruction is leveraged to store reflectance estimates and user-provided constraints in 3D space to inform the ill-posed intrinsic video decomposition problem. Our approach runs at real-time frame rates, and we apply it to applications such as relighting, recoloring and material editing.

Abstract—We present a novel real-time approach for user-guided intrinsic decomposition of static scenes captured by an RGB-D sensor. In the first step, we acquire a three-dimensional representation of the scene using a dense volumetric reconstruction framework. The obtained reconstruction serves as a proxy to densely fuse reflectance estimates and to store user-provided constraints in three-dimensional space. User constraints, in the form of constant shading and reflectance strokes, can be placed directly on the real-world geometry using an intuitive touch-based interaction metaphor, or using interactive mouse strokes. Fusing the decomposition results and constraints in three-dimensional space allows for robust propagation of this information to novel views by re-projection. We leverage this information to improve on the decomposition quality of existing intrinsic video decomposition techniques by further constraining the ill-posed decomposition problem. In addition to improved decomposition quality, we show a variety of live augmented reality applications such as recoloring of objects, relighting of scenes and editing of material appearance.

Index Terms—Intrinsic video decomposition, reflectance fusion, user-guided shading refinement

1 Introduction

The ability to edit the appearance of the real world seen through a mobile device or a head-mounted see-through display – such as photo-realistic recoloring and relighting of real scenes – is essential for many augmented reality (AR) applications. Imagine a virtual refurnishing application that allows a user to roam around and explore different color choices for real-world objects, or different placements of virtual lights, directly in their living room.

To enable this, an AR system needs to jointly track its position and reconstruct the geometry of the scene – initial solutions to this hard problem exist. The much harder problem, however, is that the system needs to solve a complex inverse rendering problem in real time. Ideally, from monocular or RGB-D sensors alone, the AR device has to estimate detailed models of surface reflectance and scene illumination, in order to modify both through computer graphics overlays. As of today, estimating *fine-grained light transport models* in *general uncontrolled scenes* from *only one on-board camera* is far from possible in *real time*.

- ** A. Meka and G. Fox assert equal contributions.*
- *A. Meka, M. Zollhöfer, and C. Theobalt are with the Graphics, Vision and Video Group at MPI for Informatics, 66123 Saarbrücken, Germany. E-mail: {ameka, mzollhoef, theobalt}@mpi-inf.mpg.de.*
- *G. Fox is with the Saarbrücken Graduate School of Computer Science, Saarland University, 66123 Saarbrücken, Germany. E-mail: fox@depend.uni-saarland.de.*
- *C. Richardt is with the University of Bath, Bath BA2 7AY, UK, and the Graphics, Vision and Video Group at MPI for Informatics. E-mail: christian@richardt.name.*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

Meka et al. [25] recently showed that intrinsic RGB video decomposition [5, 39], a simpler yet still highly complex inverse rendering problem, is feasible in real time. Making the simplifying assumption of only diffuse surfaces, intrinsic decomposition factors the video, per pixel, into its reflectance and shading components, enabling their independent modification. However, intrinsic decomposition is ill-posed [2], as the separation of reflectance color and illuminant color is ambiguous. On heavily textured objects, this inherent ambiguity leads to reflectance texture information being erroneously 'baked' into the shading. In addition, high-frequency shading effects are often misinterpreted as texture. Although regularization of reflectance and shading can reduce such artifacts, they cannot be entirely resolved in the existing techniques. This leads to visual artifacts in the targeted AR applications.

To alleviate this problem and enable live realistic editing of reflectance and lighting in augmented reality, we propose a novel interactive scene-level approach for real-time intrinsic decomposition of static scenes captured by an RGB-D sensor. Our approach is based on a volumetric representation of the scene that serves as a proxy to store reflectance estimates and sparse user-provided constraints, such as constant shading and constant reflectance strokes, directly in 3D. This has several fundamental advantages compared to 2D-based approaches. Since the user constraints are stored in 3D space, they can be robustly re-projected to arbitrary novel views of the scene to further constrain the intrinsic decomposition problem. Similarly, we densely fuse surface reflectance estimates into the volumetric reconstruction of the scene. This enables the re-projection of the estimates to novel views to further constrain and jump-start the intrinsic decomposition process. This also leads to more temporally stabilized decomposition results as demonstrated in Sect. 7. Another benefit of fusing reflectance estimates is that we obtain complete colored 3D models that are devoid of shading information.

The user constraints are intuitively provided directly on the real-world geometry with a touch-based interaction metaphor that allows

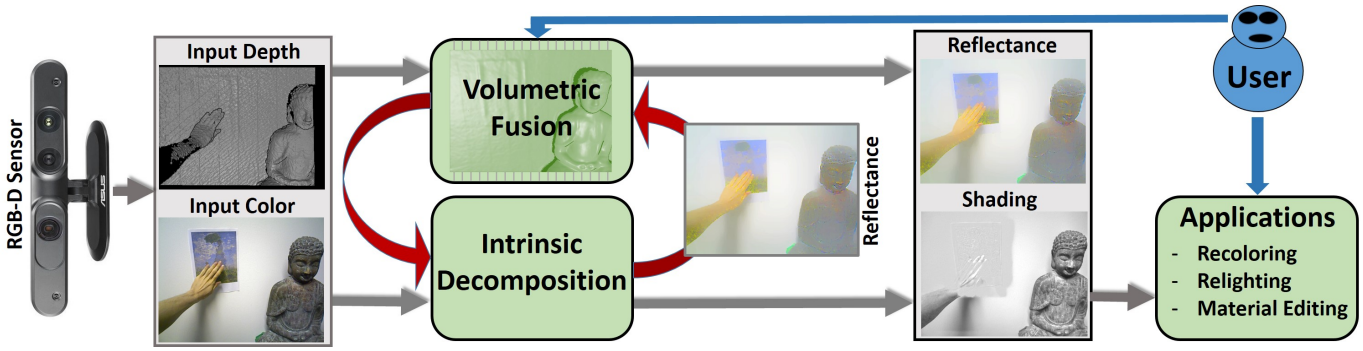

Fig. 2. Our novel user-guided intrinsic video approach enables real-time applications such as recoloring, relighting and material editing.

to define strokes on the geometry or via live mouse interactions. By providing shading and reflectance strokes, the user can interactively improve the decomposition result to resolve ambiguities and obtain higher quality results than with previous fully-automatic approaches.

The obtained decomposition quality improves on Meka et al.'s real-time approach [25], which does not consider user input and thus suffers particularly in highly textured regions. Since our approach runs live, the user can reexamine the decomposition result at any time, and place additional strokes if required. In addition to photorealistic recoloring and material editing, the availability of the underlying geometry model, which we jointly recover, enables advanced editing effects, such as physically correct relighting of objects. Note that our approach can also be used to decompose standard RGB images, without geometry, in real time, since we can also directly use 2D mouse strokes as constraints. The motivation of our work is not interaction design itself, but the design of the algorithms that enable the use of scene geometry and user interaction for enhanced intrinsic decomposition of a live video stream as well as its photorealistic augmentation. In summary, our method is based on the following main technical contributions:

- A volumetric scene representation to densely store the obtained reflectance estimates, and user-provided strokes for constant shading and reflectance in 3D.
- A real-time intrinsic decomposition approach that exploits these constraints to solve the ill-posed decomposition problem.
- Prototype augmented reality applications such as live recoloring, material editing and relighting at high quality.

2 Related Work

Intrinsic image decomposition has a long history, stretching from the seminal Retinex approach [22] all the way to the current day. The key insight of Retinex, which helps to disambiguate shading and reflectance, is that larger image gradients mostly correspond to gradients in reflectance rather than shading. Therefore, thresholding the image gradients can be used for disambiguating these two components. This idea has been extended by using learned classifiers instead of fixed thresholds [35], it has also been combined with non-local cues for improving decompositions [13, 32], and closed-form solutions have been proposed [41]. To further improve the quality of intrinsic decompositions, additional, increasingly complex priors have been proposed to constrain the solution space. Many techniques assume that the reflectance distribution in a scene is sparse [10, 13, 32, 33], i.e. that there are only a few different colors visible at the same time, which can for example be determined using clustering [12], efficient inference via dense conditional random fields [3] or image flattening [4]. Barron et al. [1] even model and estimate shape and illumination in addition to reflectance, and Kong and Black [19] also estimate contours, depth and optical flow from videos. Recently, more advanced reflectance priors have been learned directly from ground-truth intrinsic decompositions [43, 45]. Image sequences can also provide temporal constraints, for example when reflectance is constant but shading varies temporally [20,21,24,38]. The additional depth channel captured by consumer depth cameras has also been exploited to provide additional constraints [8, 16, 23]. We also use a depth camera for enabling scene-consistent temporal propagation of reflectance estimates and user constraints that are densely stored in the reconstructed scene geometry.

In many cases, the existing priors fail to achieve intrinsic decompositions of high quality. Annotations such as scribbles provided by a user can help to guide the intrinsic decomposition towards the desired solution [6, 31]. Previous approaches for off-line intrinsic video decomposition also make use of scribbles to obtain higher quality decomposition results. Ye et al. [39] use a scribble-based technique for decomposing the first video frame, and Bonneel et al. [5] allow strokes for any video frame and use them as necessary. In contrast, Meka et al.'s real-time, live intrinsic video technique [25] explicitly excludes scribbles as they cannot be provided in real time at 30 Hz. In our work, we show how to make scribbles work in a live setup by embedding them in a dense volumetric 3D reconstruction of the scene that makes them independent of the current camera viewpoint. In addition, we use the reconstruction as a proxy to fuse and temporally propagate reflectance estimates.

Placing virtual objects into a real-world scene in a seamless, photorealistic fashion requires an accurate estimate of the incident scene illumination, so that the virtual object can be lit plausibly. Gruber et al. [14] use spherical harmonics from the color observations of a jointly reconstructed 3D scene model, which enables plausible illumination effects in a virtual-reality rendering pipeline. Follow-up work extended this approach to dynamic real-world scenes within an augmented-reality rendering pipeline [15].

Scene reconstruction based on commodity RGB-D sensors employs spatiotemporal filtering [28] or implicit surface representations [9, 11, 42], since they allow to deal with the noisy depth data captured by commodity sensors. The first online method for the reconstruction of a static scene using a hand-held depth sensor was KinectFusion [17, 26]. The scene's geometry is approximated using a truncated signed distance field [9], into which per-frame data is fused. Camera tracking is based on a fast variant of the iterative closest point (ICP) algorithm [30] that uses projective correspondences. Many approaches that extend KinectFusion have been proposed, with the focus on extending the scale of the limited reconstruction volume [7, 26, 29, 34, 40]. Another extension performs an illumination-invariant 3D reconstruction [18] using time-of-flight RGB-D cameras: illumination-independent surface reflectance is first computed in the infrared domain, and then transferred to the color domain by solving an optimization problem. Our approach enables the placement of user constraints via interactions in real time, which allows to incrementally improve the decomposition results, and alleviates the texture copy problem. The SemanticPaint [36] approach combines dense volumetric reconstruction with a learning-based segmentation strategy to obtain a semantic decomposition of the scanned scene.

3 Method Overview

A high-level overview of our novel user-guided intrinsic video approach is shown in Fig. 2. First, we reconstruct a volumetric representation of the scene using the RGB-D data captured by a commodity depth sensor. To this end, we employ a dense volumetric 3D reconstruction approach [27] that obtains high-quality reconstructions of static scenes in real time (Sect. 4). In contrast to Nießner et al. [27], we use this representation as a proxy to densely fuse surface reflectance estimates instead of the input image colors. For this, we simultaneously compute an intrinsic decomposition of the color video stream during volumetric reconstruction, and fuse the obtained reflectance estimates. The fused reflectance information is used to further inform the underlying intrinsic video decomposition problem. In addition to the surface reflectance, we also store user-provided constraints in the form of constant reflectance and shading strokes. Such constraints can be provided using live mouse input or using an intuitive touch-based interaction metaphor. In the case of touch-based input, the user is automatically detected by a foreground segmentation approach that utilizes the difference in geometry and color between the reconstructed model and the current input RGB-D frame. This allows us to detect touch-based user interaction on real-world geometry, and enables the user to interactively place constraints in the scene (Sect. 5). We project these constraints into novel views to further constrain the ill-posed intrinsic decomposition problem (Sect. 6). Our proposed approach facilitates a variety of augmented reality applications, such as recoloring, material editing and relighting (Sect. 8).

4 Volumetric Reflectance Fusion

As the user walks around a scene with an RGB-D camera, which could for example be integrated into a head-mounted AR device, we obtain a virtual model of the scene using VoxelHashing, a large-scale dense volumetric reconstruction technique [27]. The source code for the VoxelHashing framework is publicly available[1]. The captured depth maps are fused into a high-quality model using a truncated signed distance field [9] (4 bytes per voxel). Memory is managed based on a space and time efficient spatial hashing strategy. Internally, 3D space is discretized into a set of discrete voxels, which are stored as blocks consisting of $8^3 = 512$ voxels each. We track the camera's rigid motion using a fast variant of the iterative closest point (ICP) algorithm that uses projective correspondence lookups.

In contrast to Nießner et al. [27], we do not fuse the observed color samples in the volume, but directly fuse surface reflectance estimates (12 bytes per voxel). To this end, we decompose the simultaneously captured color image into its shading and reflectance layers (Sect. 6). The surface reflectance is devoid of illumination information and is fused using temporal exponential averaging. Since multiple per-frame reflectance estimates are averaged, sensor noise and inconsistencies in the decomposition results are significantly reduced. In addition to surface reflectance, we use the dense volumetric reconstruction to store additional user-provided constraints based on a stroke identifier (1 byte per voxel). Storing the constraints directly in 3D world space allows us to re-project them to arbitrary novel camera views, and hence help to solve the ill-posed intrinsic decomposition problem. In addition, we exploit the spatial neighborhood information encoded in the volumetric grid to propagate constraints in 3D space (see Sect. 5.2) to obtain a basic segmentation of the scene. This is useful for applying constraints directly to larger parts of the scene, and is used in several proposed AR applications (see Sect. 8).

5 Live User Interaction

After reconstruction of the scene's geometry, the user can interact with the scene using live mouse input or a touch-based interaction metaphor to provide constraints to further inform the ill-posed intrinsic decomposition problem. Constraints are given in the form of constant reflectance and constant shading strokes. We use the dense volumetric reconstruction of the scene as a proxy to store the constraints directly in world space on a per-voxel level (using a stroke identifier attribute). For example, the user can place a constant shading stroke on a wall to alleviate the texture copy problem encountered in previous approaches, where high-frequency reflectance is often erroneously copied to the shading layer. In addition to these constraints, the user input is used in the proposed live AR applications (see Sect. 8), where it enables recoloring, material editing and relighting. The user can for example simply touch a chair to assign a different color to it, or change the material of any object in the scene. In summary, all supported interactions are:

- **Constant Shading Stroke:** All surface points belonging to the same stroke are enforced to share the same shading value. Multiple independent strokes of this type can be used.
- **Constant Reflectance Stroke:** This constraint enforces all associated surface points to share the same reflectance color. Multiple independent strokes of this type can be defined.
- **Recoloring Stroke:** The reflectance of all associated surface points is set to a fixed user-specified color. Using this stroke type, users can paint an arbitrary reflectance map.

All strokes optionally support a region filling strategy that allows to directly select a complete subset of the scene. The propagation of stroke attributes is based on spatial connectivity and reflectance similarity, as detailed below.

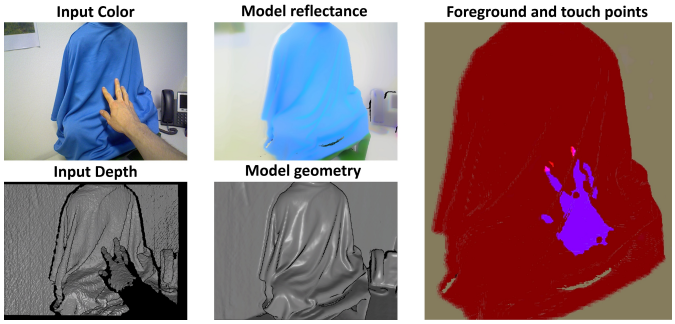

Fig. 3. The user's hand is detected as foreground based on the difference between the input depth image and the reconstructed scene. Touch points are detected (bright red) and propagated based on spatial connectivity and reflectance similarity.

5.1 Detection of Touch Points

Once scene reconstruction is finished, the integration of further geometry is stopped to allow the user to interact with the obtained reconstruction of the scene by placing constraints. Interactions are based either on live mouse input or a touch-based interaction metaphor. Touch-based interaction requires the user to closely interact with objects that are in plain view of the camera (see Fig. 3). This might throw off the camera tracker, since the motion of the user violates the assumption that the scene is static. To alleviate this problem, we automatically detect the pixels that correspond to the user and exclude them from tracking. To this end, we prune correspondences in the ICP alignment strategy if the distance between points is larger than $\varepsilon_{\text{dist}} = 15$ cm or the normals deviate more than $\varepsilon_{\text{norm}} = 14$ degrees. After alignment, all outliers in the input depth map are considered foreground. In the next step, we determine touch points based on the spatial proximity of the background and skin-colored foreground pixels [37]. For every detected touch point, we mark all voxels that fall within a small spherical neighborhood, similar to a 3D brush. The radius of the sphere can be controlled by the user. In the case of live mouse input, we use the rendered depth map to back-project the strokes to 3D space.

5.2 Spatial Constraint Propagation

To enable fast and convenient editing, we provide the user the option to automatically propagate constraints to appropriate spatial subregions.

[1] https://github.com/niessner/VoxelHashing

Semantic segmentation is a challenging and ill-posed problem, especially at real-time rates. The SemanticPaint approach [36] presents an impressive solution to this problem, but has a high memory footprint and is already quite computationally demanding. Since the goal of our approach is user-guided intrinsic decomposition, we use our own lightweight segmentation approach, which leaves enough computational resources for the other processing steps. For each stroke, we compute the average reflectance value of all influenced voxels and store it for further processing. We then perform a data-parallel flood fill to all neighboring voxels that have a sufficiently similar reflectance in RGB color space to the stored value.

The data-parallel flood fill works by propagating a 3D voxel frontier starting from a seed point. We manage the current frontier using an array in global device memory that implements a list of voxels. The insertion of new elements into the list is managed using an atomic counter. Given the current frontier, we advance the frontier in space by starting one thread per voxel in the list. Each thread examines its $3\times3\times3$-voxel neighborhood. We use a binary mask to store which voxels have already been processed, and append unprocessed neighboring voxels that fulfill the flood fill criterion to a new voxel frontier list. In the flood fill criterion, we threshold reflectance similarity based on the distance in RGB color space ($\varepsilon_{\text{fill}}=0.1$) between the reflectance of the currently processed voxel and the stored average reflectance of the current stroke. Note that the flood fill implicitly takes the connectivity of the sparse voxel grid into account.

6 Scene-Level Intrinsic Video Decomposition

The majority of intrinsic video decomposition techniques suffer from the texture copy problem, leading to residual texture in the shading layer. This is because it is highly challenging to correctly disambiguate texture into its reflectance and shading components in the absence of additional constraints. A number of intrinsic decomposition techniques have therefore resorted to user interaction in the form of strokes to provide additional constraints [5, 6, 31, 39]. We propose to use live mouse interactions or a touch-based interaction metaphor directly in 3D space for intuitive editing of the intrinsic decomposition. User input is stored densely based on the obtained scene reconstruction. In addition, we fuse computed reflectance estimates using the volume. At run time, reflectance estimates and constraints are projected to novel views to constrain the ambiguous intrinsic decomposition problem towards a higher quality solution. Previous constraint-based approaches run offline and require long computation times. In contrast, our approach runs at real-time frame rates, thus making it usable in the augmented reality context.

6.1 Variational Intrinsic Video Decomposition

We cast finding the optimal intrinsic decomposition $\mathbf{D}^*$ as a general non-linear energy minimization problem:

$$\mathbf{D}^* = \underset{\mathbf{D}}{\operatorname{argmin}} E(\mathbf{D}) \tag{1}$$

$$\mathbf{D} = \left[\ldots, \mathbf{r}(\mathbf{x})^\top, \ldots, s(\mathbf{x}), \ldots\right]^\top, \tag{2}$$

where the vector $\mathbf{D}$ contains all unknowns, i.e. log-space reflectance $\mathbf{r}(\mathbf{x})\in\mathbb{R}^3$ and shading $s(\mathbf{x})\in\mathbb{R}$ for all pixels $\mathbf{x}\in\Omega\subset\mathbb{N}^2$. The employed intrinsic video decomposition energy E is based on several objective functions:

$$E(\mathbf{D}) = E_{\text{fit}}(\mathbf{D}) + w_{\text{r}}E_{\text{reg}}(\mathbf{D}) + w_{\text{u}}E_{\text{user}}(\mathbf{D}) + w_{\text{s}}E_{\text{stab}}(\mathbf{D}). \tag{3}$$

The objective functions model the reproduction of the input image E_{fit}, spatio-temporal regularization E_{reg}, integration of the user constraints E_{user}, and temporal stabilization E_{stab}. The constant weights $w_{\text{r}}=1$, $w_{\text{u}}=1000$ and $w_{\text{s}}=10$ control the influence of the different objectives. In the following, we discuss all terms in more detail.

Reproduction of the Input Image The fitting objective E_{fit} enforces that the decomposition reproduces all N pixels of the input color image $\mathbf{I}$. We formulate this constraint in the log-domain for linearity:

$$E_{\text{fit}}(\mathbf{D}) = \sum_{\mathbf{x}\in\Omega} \left\| \mathbf{i}(\mathbf{x}) - \left(\mathbf{r}(\mathbf{x}) + [1\ 1\ 1]^\top s(\mathbf{x})\right)\right\|^2. \tag{4}$$

Here, $\mathbf{i}(\mathbf{x})=\ln\mathbf{I}(\mathbf{x})\in\mathbb{R}^3$ is the logarithm of the pixel color at pixel $\mathbf{x}$, $\mathbf{r}(\mathbf{x})=\ln\mathbf{R}(\mathbf{x})$ is the log-reflectance, and $\mathbf{s}(\mathbf{x})=\ln\mathbf{S}(\mathbf{x})$ the log-shading value of the same pixel.

Spatio-Temporal Regularization For regularization, we follow the approach of Meka et al. [25] and employ a combination of four different terms:

$$E_{\text{reg}}(\mathbf{D}) = w_{\text{p}}E_{\text{p}}(\mathbf{D}) + w_{\text{g}}E_{\text{g}}(\mathbf{D}) + w_{\text{m}}E_{\text{m}}(\mathbf{D}) + w_{\text{c}}E_{\text{c}}(\mathbf{D}). \tag{5}$$

Since many man-made scenes contain a small, distinct number of reflectance values, we enforce sparsity based on a p-norm constraint:

$$E_{\text{p}}(\mathbf{D}) = \sum_{\mathbf{y}\in\mathbf{N}(\mathbf{x})} \omega_{\text{cs}}(\mathbf{x},\mathbf{y})\cdot\|\mathbf{r}(\mathbf{x})-\mathbf{r}(\mathbf{y})\|_2^p, \tag{6}$$

where $\omega_{\text{cs}}(\mathbf{x},\mathbf{y}) = \exp(-15\cdot\|\mathbf{c}(\mathbf{x})-\mathbf{c}(\mathbf{y}))\|_2)$ measures the chroma similarity of two adjacent pixels. Spatio-temporal coherence is incorporated based on a global prior E_{g} that takes long-range chroma similarity into account. The prior E_{m} enforces ℓ_2-spatial smoothness of the shading layer at chroma boundaries. Finally, a soft constraint on chroma similarity E_{c} keeps the chroma of the reflectance image close to the chroma of the input. For a detailed discussion of the terms $E_\bullet$, the sparsity norm ℓ_p and parameters $\omega_\bullet$, we refer to Meka et al. [25].

Incorporation of User Constraints One of the main contributions of our work is a novel approach for incorporating the user constraints, in the form of constant reflectance and constant shading strokes, directly into the optimization problem:

$$E_{\text{user}}(\mathbf{D}) = \sum_{\mathbf{S}_i\in S}\sum_{\mathbf{x}\in\mathbf{S}_i} w_i(\mathbf{x})\cdot|s(\mathbf{x})-\hat{s}_i|^2 + \sum_{\mathbf{R}_i\in R}\sum_{\mathbf{x}\in\mathbf{R}_i} w_i(\mathbf{x})\cdot\|\mathbf{r}(\mathbf{x})-\hat{\mathbf{r}}_i\|^2. \tag{7}$$

Here, S is the set of shading strokes, and $\mathbf{S}_i$ the set of pixels belonging to the i-th shading stroke. $\hat{s}_i$ is the representative unknown shading value associated with the i-th stroke. The same notation holds for the reflectance strokes $\mathbf{R}_i\in R$. Note that $\hat{s}_i$ and $\hat{\mathbf{r}}_i$ are unknown auxiliary variables. For the i-th stroke, we define a per-pixel stroke weight $w_i(\mathbf{x})$ to fade out the influence of the strokes (squared fall-off) close to their boundary.

Stabilization of Reflectance Estimates Another important contribution of our work is to densely fuse the obtained reflectance estimates into the volumetric scene representation. This allows to enforce temporal coherence and further inform the intrinsic decomposition problem. To this end, we propose the following novel stabilization constraint:

$$E_{\text{stab}}(\mathbf{D}) = \sum_{\mathbf{x}\in\Omega} B(\mathbf{x})\cdot\left(\mathbf{r}(\mathbf{x})-\mathbf{r}^{\text{model}}(\mathbf{x})\right)^2. \tag{8}$$

The background mask $B(\mathbf{x})$ (one for background, zero for foreground) prunes any potentially dynamic foreground pixels. It encourages the per-pixel log-reflectance values $\mathbf{r}(\mathbf{x})$ to be close to the fused mean log-reflectance $\mathbf{r}^{\text{model}}(\mathbf{x})$ stored in the volumetric scene model. This per-pixel mean log-reflectance is computed by extracting the log-reflectance-colored isosurface of the volumetric scene representation via ray marching.

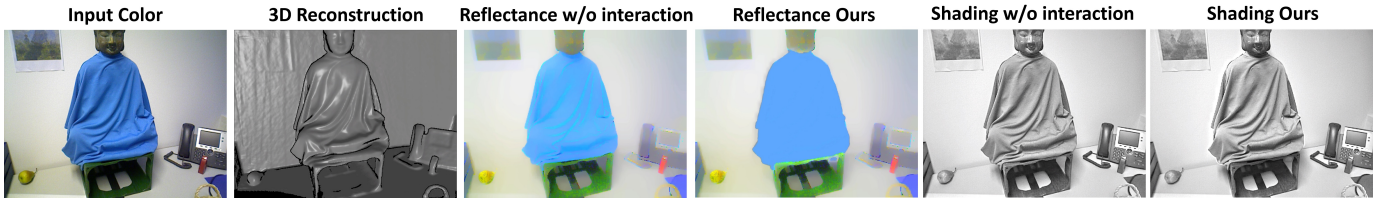

Fig. 4. Constant reflectance strokes improve the decomposition by moving the high-frequency shading of the cloth to the shading layer.

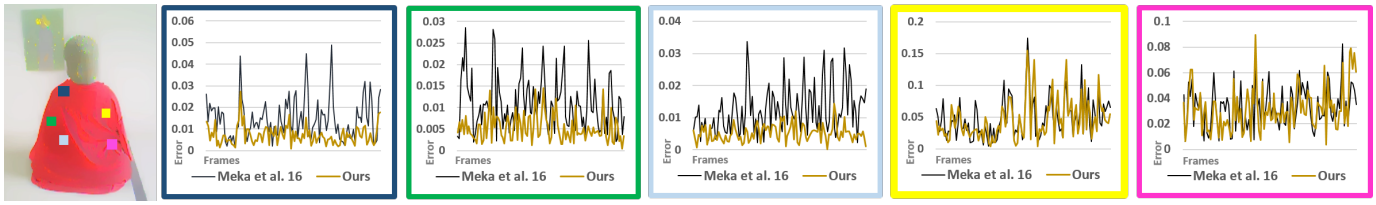

Fig. 5. Temporal reflectance constancy. We track five rectangular regions and compute the average albedo difference over time per region. Our approach uses fused reflectance estimates to further constrain and jump-start the intrinsic decomposition process. Therefore, it obtains a higher temporal reflectance consistency than the approach of Meka et al. [25].

6.2 Data-Parallel Optimization

Our goal is to compute the intrinsic decomposition of the RGB-D video stream at real-time frame rates. We therefore require a fast and efficient strategy to solve the underlying non-linear optimization problem. We propose a highly efficient, data-parallel, iteratively reweighted least squares (IRLS) solver that allows computing the optimum of the energy E (see Equation 3) at real-time rates. In contrast to Meka et al. [25], our decomposition objective does not have a sparse Jacobian, but contains dense blocks due to the incorporation of the user constraints E_{user} (see Equation 7). This is because every per-pixel unknown belonging to the same stroke has a derivative with respect to the unknown per-stroke auxiliary variable. Therefore, the data-parallel solver proposed in previous work is not sufficient to achieve real-time frame rates, since the workload is not equally distributed between different threads. Such a joint optimization of all variables would lead to faster convergence with respect to the required number of iteration steps, but every iteration of the solver would require significantly more time, since the data-parallel compute power of the GPU can not be fully exploited.

To tackle this problem, we propose an iterative flip-flop strategy that solves two simpler optimization problems in alternation. Given an initial estimate of the per-pixel shading and reflectance, we first optimize for the auxiliary variables. Since the auxiliaries only appear in the least-squares objective E_{user}, the optimum has a closed-form solution, and can be obtained as the average of all associated per-pixel values that belong to the same stroke. After this, we fix the new values for the auxiliaries, and optimize for a new decomposition using a data-parallel IRLS solution strategy. As the auxiliaries are constant during this step, the Jacobian is again sparse, leading to high performance. We assume convergence after 7 iteration steps. Internally, the IRLS solver divides the problem into small rectangular sub-domains and uses a data-parallel variant of the alternating Schwarz procedure [44]. Each iteration solves the local problems in shared memory and exchanges data with neighboring domains using global memory. We apply this optimization using a coarse-to-fine strategy (5 levels) for faster convergence. Starting from the coarsest level, we solve the coarse scale version of the problem to obtain an approximate solution. We then upsample this solution to the next finer level and use it for initialization.

7 Results

We demonstrate our approach in a live setup. We use a PrimeSense Carmine 1.09 close-range RGB-D sensor to obtain two 640×480 video streams of color and depth at 30 Hz. Note that our approach is agnostic to the specific RGB-D sensor being used. We only require a spatially and temporally aligned color and depth stream as input. After acquiring an initial geometric model of the scene using dense volumetric reconstruction, the decomposition quality is improved using strokes that enforce constant reflectance and shading. We make our test scenes, including the results obtained by our approach, publicly available on our project page[2] to encourage follow-up work and enable others to easily compare to our approach.

Decomposition Results Intrinsic decomposition of a scene allows for independent modification of the underlying physical layers of a scene while preserving the photorealism on reconstruction. Even for simple scenes with uniform, untextured regions, such photorealism cannot be obtained by simple luminance–chrominance decomposition due the problem of 'chromaticity shift', as described by Meka et al. [25]. For textured surfaces, current state-of-the-art intrinsic decomposition approaches suffer from the texture-copy problem, if they do not rely on additional user input. Texture-copy refers to texture variation being misinterpreted as shading, as in Fig. 6 (top). Our approach allows to resolve this problem via the incorporation of constant shading strokes into the decomposition problem, see Fig. 6 (bottom). Without user input, it is difficult to disambiguate between blocks of varying intensity, and current state-of-the-art approaches fail in this regard. By adding user constraints, the optimization approach better resolves the inherent ambiguities of the intrinsic decomposition problem, and we obtain a cleaner shading layer.

In addition, the decomposition of uniformly colored regions suffers from the previously mentioned chromaticty-shift problem due to high-frequency shading variation, which is easily improved using a constant reflectance stroke, as can be seen in Fig. 4. The clothing contains several dark creases that wrongly end up in the reflectance layer in the absence of interaction. With an appropriate stroke, directly on the 3D geometry, our approach mitigates this issue and ensures constant reflectance over the cloth.

Runtime Performance and Memory Requirements For the reconstruction of the static scene geometry, we use a voxel resolution of 1 cm. Camera tracking takes 2 ms and reflectance fusion 0.6 ms. To project the user constraints into the image, we use ray marching, which takes 14 ms to compute the stroke map. Overall, our scene-level intrinsic decomposition runs at real-time frame rates. We use 5 hierarchy levels with 7 IRLS iterations on each. Each non-linear IRLS iteration performs 7 PCG steps internally. After each non-linear iteration, we perform a flip-flop step to update the auxiliary variables (Sect. 6.2). Intrinsic decomposition takes in total 22 ms per frame. While performing the 3D reconstruction of the scene, we achieve an average frame rate of 25 Hz.

[2] http://gvv.mpi-inf.mpg.de/projects/InteractiveIntrinsicAR/

Table 1. Per-frame run time of our user-guided intrinsic video approach averaged over the entire sequence for the scene in Figure 8.

	Input	ICP	Decomposition	Fusion	Relighting
time (ms)	6.1	3.5	9.1	6.7	8.4

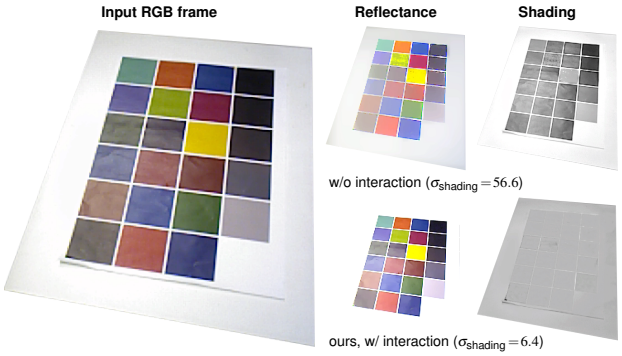

Fig. 6. Intrinsic decomposition results for a color chart. Without interaction, the shading image suffers from texture-copy. Our approach improves the decomposition by using a constant shading stroke. This reduces the intensity variation of the shading layer (smaller standard deviation $\sigma_{shading}$).

After the static reconstruction is completed, the frame rate increases to more than 30 Hz. All timings are computed on a commodity Nvidia GTX Titan graphics card. A more detailed breakdown of the timings of our approach is shown in Table 1.

Our approach has a higher memory footprint than off-the-shelf VoxelHashing [27], since we store the surface log-reflectance using four bytes per color channel. The memory footprint could be decreased by storing a discretized version of surface reflectance, e.g. one byte per color channel. For our scenes, this is not the limiting factor, since more than 12 GB of global device memory are available on modern graphics hardware. For example, the sequence shown in Fig. 8 requires in total 9.4 GB of global device memory for a voxel resolution of 4 mm (3.7 GB for a voxel resolution of 1 cm). This also includes all the data structures used during optimization and the memory to store the user constraints.

Reflectance Initialization In addition to the user constraints, we also densely fuse surface reflectance estimates using the volumetric reconstruction of the scene. This allows to project the reflectance estimates to arbitrary novel views, which can be used to further constrain and jump-start the intrinsic decomposition process. The technique of Meka et al. [25] initializes the reflectance layer in every frame with the input RGB image, which could be far from the correct reflectance values. In contrast, our approach only uses this initialization for the first frame, and for subsequent frames synthesizes an initial reflectance map based on the projection of the fused reflectance estimates to the novel view. Occluded regions are initialized based on the corresponding input RGB values. Our novel temporal stabilization term also helps to stabilize the decomposition results, see Fig. 5. We track five rectangular regions and compute the average albedo difference per region over time. As can be seen, our approach obtains a higher temporal reflectance stability than Meka et al. [25] (average norm of albedo variation: 0.0187 instead of 0.0241). We refer to the accompanying video for the complete sequence. Another benefit of fusing reflectance estimates is that we obtain a complete colored 3D model that is devoid of shading information, see Fig. 8, which is in contrast to the color reconstructed by state-of-the-art volumetric reconstruction techniques [27].

Comparison to the State-of-the-Art We compare our approach to the existing off-line intrinsic video approaches by Ye et al. [39] and Bonneel et al. [5], which also use user-provided strokes for constraining

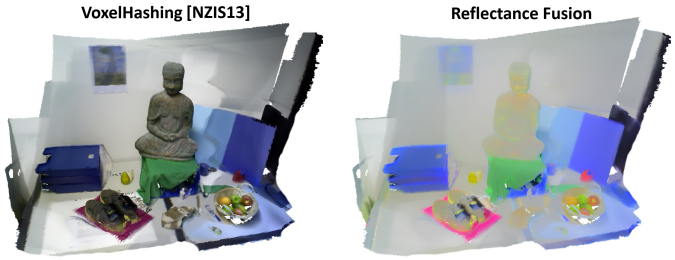

Fig. 8. Our approach reconstructs the reflectance of the scene.

Fig. 9. Photorealistic recoloring of a shirt using our approach.

the result, as well as Meka et al.'s fully automatic real-time approach [25]. Note that these techniques work in a slightly less constrained setup, with a standard RGB camera, while our approach additionally leverages the available depth information of commodity RGB-D sensors. As these approaches operate on monocular color video alone (without depth), we compare the decomposition quality on the 'girl' dataset in Fig. 7, without using any geometry reconstruction and only 2D strokes within our approach. This comparison shows that our approach obtains comparable decompositions to state-of-the-art off-line approaches [5, 39], but at real-time frame rates. Our decomposition quality improves on Meka et al.'s real-time approach which does not consider user input, especially in regions with high texture variation, such as the logo on the shirt (see inset in Fig. 7). Additional user constraints clearly help to resolve the inherent ambiguities of the intrinsic decomposition problem. Unlike existing methods, our approach works best with RGB-D video streams, as strokes are placed directly on 3D geometry and can be projected to novel views of the scene for initializing them. Since our approach runs live, the user can reexamine the decomposition result at any time, and place additional strokes if required.

8 Interactive Applications

Our method enables a wide variety of interactive applications. In the following, we show several examples, such as photorealistic recoloring, material editing and geometry-based relighting.

Photorealistic Recoloring and Material Editing We support interactive and intuitive recoloring and material editing of real-world objects. Using the presented color-based volumetric segmentation strategy, the complete geometry of the object that should be modified is first segmented. Since the segmentation is computed in 3D, we can segment the entire object, even if it is not completely visible from the current view. The selected 3D geometry is projected to novel views to obtain the 2D mask that is later on used to modify the appearance of the object. For recoloring, we replace an object's color in the computed reflectance map of the current frame's decomposition by a user-defined color. For material editing, we apply a tone-mapping filter (as used by Ye et al. [39]) on the shading layer within the mask region. The modified layer is then recombined with the other intrinsic layer to obtain the final output, see Fig. 9. By simply touching an object, the user can also choose to pick up a color from the environment. This color can then be used to recolor other objects, as illustrated in Fig. 10. Instead of modifying the reflectance layer, we can also apply a tone-mapping function to the shading layer to change the appearance of an arbitrary object's material. This enables us for example to manipulate the appearance of a plaster cast such that it looks like metal, see Fig. 12.

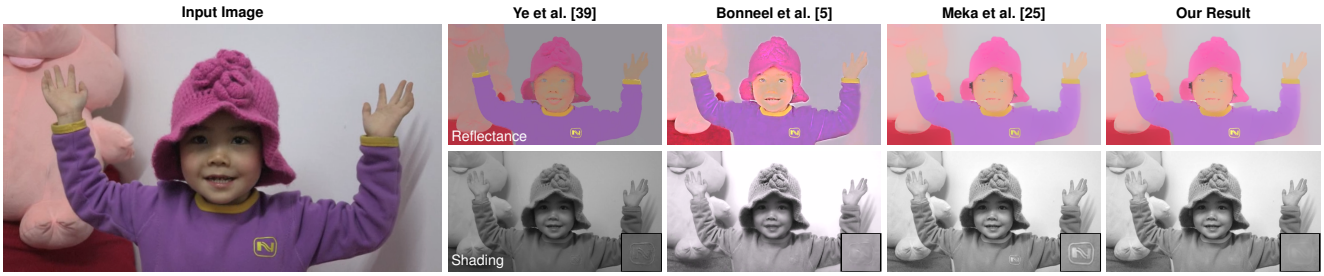

Fig. 7. Comparison to state-of-the-art intrinsic video decomposition techniques on the 'girl' dataset. Our approach matches the real-time performance of Meka et al. [25], while achieving the same quality as previous off-line techniques [5, 39] (see zooms).

Fig. 11. Dynamic geometry-based relighting. A virtual shading image is generated by rendering the scene geometry under a new light source. The resulting shading map is blended with the shading layer before recombining it with the reflectance to obtain a relighting effect.

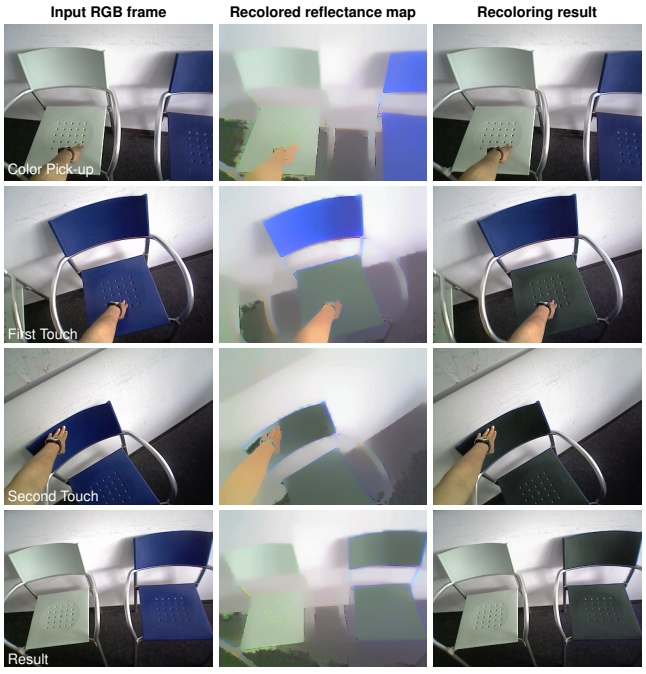

Fig. 10. Interactive object recoloring. The reflectance of the light green chair is first picked up, and then transferred to the blue chair while preserving its brightness.

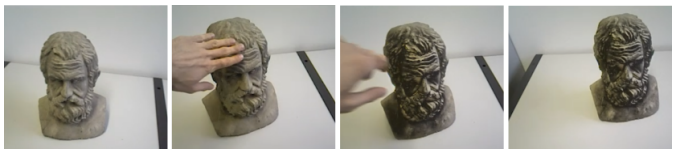

Fig. 12. We modify the shading layer to convert plaster to metal.

Geometry-Based Relighting In addition to modifications of the reflectance layer, we also present geometry-based relighting via modification of the shading layer. To this end, the user can place virtual light sources in the scene, which interact with it. We use the reconstructed 3D geometry in conjunction with the virtual light sources to generate a new shading image. The scene geometry is extracted using ray marching, and the synthetic shading map is computed by a fragment shader. We blend the shading layer of our decomposition with the synthesized shading map, then recombine the new shading layer with the reflectance map to obtain a compelling relighting effect, as shown in Fig. 11.

9 Limitations

We demonstrated that high-quality user-guided live intrinsic decomposition enables new scene modification applications. Still, our approach has a few limitations. The geometric model of the scene is currently obtained beforehand in a pre-process, since it is required as the basis for foreground/background segmentation. In the future, alternative segmentation strategies can be developed.

Our approach can only improve the decomposition quality of static scene geometry, since the user constraints are placed in 3D, and tracked based on a rigidly reconstructed scene model. Tracking dynamic time-dependent motion of non-rigidly deforming surfaces to also add and propagate constraints in such regions can be further investigated.

The improvement in decomposition quality via user constraints is of local nature, since the placed strokes only influence the decomposition result in a small surrounding neighborhood. Therefore, similar to other stroke-based approaches, a lot of such constraints might be required to completely correct an initially very erroneous decomposition result. Fortunately, this is rarely the case and constraints are only required to deal with the highly textured regions of the scene.

Our simple touch-based interaction strategy sometimes leads to erroneous detections; more robust touch detection strategies are left for future work. Constraint propagation based only on color and spatial proximity can lead to suboptimal segmentation results. This could be alleviated by the integration of a more sophisticated adaptive semantic segmentation strategy, such as SemanticPaint [36].

Similar to other intrinsic decomposition approaches that rely on user interaction, our approach assumes that the constraints provided by the user are correct. If the user provides implausible constraints, e.g. paints a constant reflectance stroke across a highly textured region, the optimization will blindly try to satisfy these incorrect constraints thus leading to unrealistic results. Guiding the user and providing some feedback regarding the satisfiability of the provided constraints is a challenging, but interesting, problem for future work.

Finally, our approach is computationally quite demanding and currently requires a state-of-the-art graphics card to achieve real-time performance. A robust, mobile and lightweight solution to the presented problem can be an enabling technology for AR devices.

10 Conclusion

We presented a novel real-time approach for user-guided intrinsic decomposition of static scenes. Users can improve the decomposition quality based on live mouse input or an intuitive touch-based interaction metaphor that allows to place decomposition constraints directly in 3D space. The constraints are projected to 2D and used to further constrain the ill-posed intrinsic decomposition problem. We also use the dense reconstruction as a proxy to fuse the obtained reflectance estimates. Our novel stabilization term applies constraints based on the projected fused reflectance estimates leading to temporally more coherent decomposition results. The intrinsic decompositions obtained by our approach show state-of-the-art quality at real-time frame rates. In addition, we demonstrated video editing tasks such as recoloring, relighting and material editing based on the obtained decompositions.

We believe that the presented live setup is the foundation for many augmented reality applications, such as virtual refurnishing, which would allow the user to explore different color and design choices for real-world objects directly in their living room.

Acknowledgments

We thank the anonymous reviewers for their helpful feedback. This work was supported by the ERC Starting Grant CapReal (335545).

References

[1] J. T. Barron and J. Malik. Shape, illumination, and reflectance from shading. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(8):1670–1687, August 2015.

[2] H. G. Barrow and J. M. Tenenbaum. Recovering intrinsic scene characteristics from images. In *Computer Vision Systems*, 1978.

[3] S. Bell, K. Bala, and N. Snavely. Intrinsic images in the wild. *ACM Transactions on Graphics*, 33(4):159:1–12, July 2014.

[4] S. Bi, X. Han, and Y. Yu. An l_1 image transform for edge-preserving smoothing and scene-level intrinsic decomposition. *ACM Transactions on Graphics*, 34(4):78:1–12, July 2015.

[5] N. Bonneel, K. Sunkavalli, J. Tompkin, D. Sun, S. Paris, and H. Pfister. Interactive intrinsic video editing. *ACM Transactions on Graphics*, 33(6):197:1–10, November 2014.

[6] A. Bousseau, S. Paris, and F. Durand. User-assisted intrinsic images. *ACM Transactions on Graphics*, 28(5):130:1–10, December 2009.

[7] J. Chen, D. Bautembach, and S. Izadi. Scalable real-time volumetric surface reconstruction. *ACM Transactions on Graphics*, 32(4):113, July 2013.

[8] Q. Chen and V. Koltun. A simple model for intrinsic image decomposition with depth cues. In *ICCV*, pages 241–248, December 2013.

[9] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *SIGGRAPH*, 1996.

[10] S. Ding, B. Sheng, X. Hou, Z. Xie, and L. Ma. Intrinsic image decomposition using multi-scale measurements and sparsity. *Computer Graphics Forum*, 2016.

[11] S. Fuhrmann and M. Goesele. Floating scale surface reconstruction. *ACM Transactions on Graphics*, 33(4):46:1–11, July 2014.

[12] E. Garces, A. Munoz, J. Lopez-Moreno, and D. Gutierrez. Intrinsic images by clustering. *Computer Graphics Forum*, 31(4):1415–1424, 2012.

[13] P. V. Gehler, C. Rother, M. Kiefel, L. Zhang, and B. Schölkopf. Recovering intrinsic images with a global sparsity prior on reflectance. In *NIPS*, 2011.

[14] L. Gruber, T. Richter-Trummer, and D. Schmalstieg. Real-time photometric registration from arbitrary geometry. In *ISMAR*, 2012.

[15] L. Gruber, J. Ventura, and D. Schmalstieg. Image-space illumination for augmented reality in dynamic environments. In *VR*, 2015.

[16] M. Hachama, B. Ghanem, and P. Wonka. Intrinsic scene decomposition from RGB-D images. In *ICCV*, 2015.

[17] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon. KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera. In *UIST*, pages 559–568, 2011.

[18] C. Kerl, M. Souiai, J. Sturm, and D. Cremers. Towards illumination-invariant 3D reconstruction using ToF RGB-D cameras. In *3DV*, 2014.

[19] N. Kong and M. J. Black. Intrinsic depth: Improving depth transfer with intrinsic images. In *ICCV*, 2015.

[20] N. Kong, P. V. Gehler, and M. J. Black. Intrinsic video. In *ECCV*, 2014.

[21] P.-Y. Laffont and J.-C. Bazin. Intrinsic decomposition of image sequences from local temporal variations. In *ICCV*, 2015.

[22] E. H. Land and J. J. McCann. Lightness and retinex theory. *Journal of the Optical Society of America*, 61(1):1–11, January 1971.

[23] K. Lee, Q. Zhao, X. Tong, M. Gong, S. Izadi, S. Lee, P. Tan, and S. Lin. Estimation of intrinsic image sequences from image+depth video. In *ECCV*, 2012.

[24] Y. Matsushita, S. Lin, S. Kang, and H.-Y. Shum. Estimating intrinsic images from image sequences with biased illumination. In *ECCV*, 2004.

[25] A. Meka, M. Zollhöfer, C. Richardt, and C. Theobalt. Live intrinsic video. *ACM Transactions on Graphics*, 35(4):109:1–14, July 2016.

[26] R. A. Newcombe, A. J. Davison, S. Izadi, P. Kohli, O. Hilliges, J. Shotton, D. Molyneaux, S. Hodges, D. Kim, and A. Fitzgibbon. KinectFusion: Real-time dense surface mapping and tracking. In *ISMAR*, 2011.

[27] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger. Real-time 3D reconstruction at scale using voxel hashing. *ACM Transactions on Graphics*, 32(6):169:1–11, November 2013.

[28] C. Richardt, C. Stoll, N. A. Dodgson, H.-P. Seidel, and C. Theobalt. Coherent spatiotemporal filtering, upsampling and rendering of RGBZ videos. *Computer Graphics Forum*, 31(2):247–256, May 2012.

[29] H. Roth and M. Vona. Moving volume KinectFusion. In *BMVC*, 2012.

[30] S. Rusinkiewicz and M. Levoy. Efficient variants of the ICP algorithm. In *3DIM*, pages 145–152, 2001.

[31] J. Shen, X. Yang, Y. Jia, and X. Li. Intrinsic images using optimization. In *CVPR*, 2011.

[32] L. Shen and C. Yeo. Intrinsic images decomposition using a local and global sparse representation of reflectance. In *CVPR*, 2011.

[33] L. Shen, C. Yeo, and B.-S. Hua. Intrinsic image decomposition using a sparse representation of reflectance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(12):2904–2915, December 2013.

[34] F. Steinbrücker, J. Sturm, and D. Cremers. Volumetric 3D mapping in real-time on a CPU. In *ICRA*, 2014.

[35] M. F. Tappen, W. T. Freeman, and E. H. Adelson. Recovering intrinsic images from a single image. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(9):1459–1472, September 2005.

[36] J. Valentin, V. Vineet, M.-M. Cheng, D. Kim, J. Shotton, P. Kohli, M. Nießner, A. Criminisi, S. Izadi, and P. Torr. SemanticPaint: Interactive 3D labeling and learning at your fingertips. *ACM Transactions on Graphics*, 34(5):154:1–17, November 2015.

[37] V. Vezhnevets, V. Sazonov, and A. Andreeva. A survey on pixel-based skin color detection techniques. In *Graphicon*, 2003.

[38] Y. Weiss. Deriving intrinsic images from image sequences. In *ICCV*, 2001.

[39] G. Ye, E. Garces, Y. Liu, Q. Dai, and D. Gutierrez. Intrinsic video and applications. *ACM Transactions on Graphics*, 33(4):80:1–11, July 2014.

[40] M. Zeng, F. Zhao, J. Zheng, and X. Liu. Octree-based fusion for realtime 3D reconstruction. *Graphical Models*, 75(3):126–136, May 2013.

[41] Q. Zhao, P. Tan, Q. Dai, L. Shen, E. Wu, and S. Lin. A closed-form solution to Retinex with nonlocal texture constraints. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(7):1437–1444, July 2012.

[42] Q.-Y. Zhou and V. Koltun. Dense scene reconstruction with points of interest. *ACM Transactions on Graphics*, 32(4):112:1–8, July 2013.

[43] T. Zhou, P. Krähenbühl, and A. Efros. Learning data-driven reflectance priors for intrinsic image decomposition. In *ICCV*, 2015.

[44] M. Zollhöfer, A. Dai, M. Innmann, C. Wu, M. Stamminger, C. Theobalt, and M. Nießner. Shading-based refinement on volumetric signed distance functions. *ACM Transactions on Graphics*, 34(4):96:1–14, July 2015.

[45] D. Zoran, P. Isola, D. Krishnan, and W. T. Freeman. Learning ordinal relationships for mid-level vision. In *ICCV*, 2015.